# Context: An Information Medium for Dynamic and Collaborative Situations

Romain Bénard, Matthieu Aubry and Pierre De Loor
CERV European Center for Virtual Reality
25, rue Claude Chappe
BP 38 F-29280 Plouzané (France)
email :{benard, maubry, deloor}@enib.fr

## Abstract

*We introduce, in this article, a way for building a Virtual Environment for Training to manage dynamic and collaborative situations in team sport. In these situations, relevant information can not be totally known in advance. Autonomous agents can only reason with few pieces of information and explanations on the situation are numerous and not easily predictable. We introduce, in this paper, the usefulness of the notion of context, in a VET, to set up agent decision-making and to generate explanations in order to help the learner. The main purpose of this article is that the context is a medium facilitating information reuse.*

## 1 Introduction

We introduce, in this article, tools and models for a constructivist approach of competences learning. We define competences as an association between knowledges and know-how. The constructivist paradigm is based on the idea that the training can be done by action [1]. The learner creates its own representations by carrying out exercises. For economic, technological or human reasons, it is relevant to set up this type of training by means of simulations thanks to virtual reality.

This gave rise to tools called Virtual Environments for Training [2]. These tools not only simulate exercices and interactions with the problem to learn but also guide learning by various means (sounds, brightness, obliteration). Within this framework, some approaches are focused on collective human activities. It is then necessary to simulate the behavior of human activities and human collaboration with the learner. Initially, research has been concentrated on procedural work. In this case, whole exercise can be described in the form of a scenario [3, 4].

Our work is distinguished from these previous approaches because it relates to the training of Dynamic and Collaborative Situations. Those can be found in various domains of teamwork with time pressure (rescue, security) [5, 6] or in team sport [7]. This implies various protagonists that interact in a common environment and have to solve a problem. Protagonists autonomy and situations varieties prevent from an exhaustive description of exercises and decision-making [6, 8].

Indeed, relevant information for the decision-making, the simulation and the training are contextual. There are behavioral schemes characterizing of the episodes of exercises but not a foreseeable single scenario. We will show how we model this contextual information and how we reuse them for three different activities:

- Development of a decision-making engine for virtual human, based on context-based reasoning [9].
- Development of the follow-up of strategic episodes allowing to evaluate what the learner does and his next action.
- Development of a contextual teaching help.

After a presentation of our context modeling, each of the three points below is the subject of one of the 3 other sections of this paper. An example of assistance is presented before the conclusion.

## 2 Context modelling and use

### 2.1 Definition

We argue as [10, 11], that context can be a way to share knowledges and information, we introduce here how we model context and use it to understand dynamic and collaborative situations.

In [12], we have introduced our vision of a global context for a virtual agent of our simulation, as well as the division of this context in sub-contexts. This splitting up allows to emphasize contexts used for decision-making and those used for explanations generation.

We describe here the structure of our context and how we can use information stored in it. A context is the sum of perceptions. Those one can be either every perception of an autonomous agent or only relevant perceptions defined by an expert.

In the first case, the agent has a library of perceptions types already defined such as *Presence*, *Distance*, *HasTheBall*... It will try to find them in the simulation. A player *A* viewing a player *B* will have in its context *Distance{A,B}*, *Presence{B}*...

In the second case, context is the sum of relevant perceptions to take a decision. We will see later how it is used in graph or for decision-making. An example of a context for doing a pass can be just seeing a partner which is unmarked and stands at a reasonable distance. The next part describes the structure of a perception.

### 2.2 Perception definition

A perception is characterized by a range of value called *value_range* that one can find for this perception. A *value* is associated to the perception. Thanks to the value and the range of value, one can compare two perceptions values. This point is detailed in the next section. A perception has a *source* and a *target* which allow to precise objects which are perceived.

We have added for each perception another attribute which is called *pedagogical_feature* that allows us to represent this perception in our environment. This feature can be one or more graphical primitives. We will detail this point in the example of this paper. One has to know that this allows to emphasize the perception.

To illustrate how this structure works, we will show the example of a perception, coming from our simulation, representing the distance between a player *A* and a player *B*. We will illustrate how this information can be reified.

| Distance |
|---|
| value_range: *nearest, near, far* |
| value: *far* |
| target: *A* |
| source: *B* |
| pedagogical_feature: *arrow* |

One can see on this example, that it is really easy to retrieve the piece of information that the agent has used to take a decision. One can already foresee the usefulness of this information representation to set up an explanatory system.

Grouping perceptions of an agent will give the necessary information on its personnal context. We will now detail two ways to use a context and how we can use information stored in contexts.

### 2.3 Using context and its information

In order to define the decision-making engine of our autonomous agents, we are using the context-based reasoning paradigm [9] which is based on the case-based reasoning introduced by [13]. These paradigms allow to reuse past experience to solve a new problem. To do that, an agent can look for in a case base in which all previous solved cases (or contexts) are stored. It will look for the most similar case present in the base to adapt it to the current situation. The past experience of the agent will be used to grow the base of actions corresponding to a context. This mechanism, described in the next section, uses the information contained in the context to choose the best action.

The second way of using context in our environment is the setting up of a contextual help. We choose to use the notion of contextual graph introduced in [14], and we have expanded in order to make them usable with dynamic situations, as we will detail in section 4. Thanks to contextual graphs, we can represent a complete strategy. We are using the graphs defined to follow the learner evolution at important moments of the simulation. If the strategy is known, pedagogical agents can set up relevant helps for the learner using the information contained in the context. The last part of this paper shows this type of assistance and the possibilities offered with graphs use.

## 3 Context-based reasoning

### 3.1 CxBR cycle

A CxBR cycle can be divided in four steps that are illustrated on figure 1. The first step consists in building the context corresponding to the current situation. This context represents every perception of the agent. The next step corresponds to the similarity measurement between the current context and those stored in the base. The adaptation step consists in adjusting parameters of the selected action to make them correct according to the current situation. The last step is the memorization process, which allows to store new solutions in the agent case base. This will enhance the agent experience during the simulation.

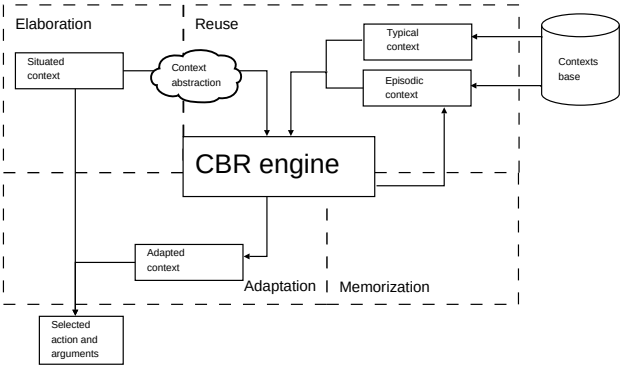


**Figure 1. A CxBR Engine**

### 3.2 Similarity measurement

The similarity function allows to identify the closest context, it is a crucial point in our reasoning engine. In our case, two contexts can be compared by calculating the similarity between every perception of the context previously solved and the perception of the current context. As in the CBR paradigm, the case that should be solved is called *target* and the context to compare is called *source*. This similarity is a function bounded between 0 and 1. A similarity equals to 1 indicates that the two cases are equals. This function looks like:

$$Sim(C_s, C_t) = 1 - \sum_{i=1}^{n} w_i * |(p_i^s - p_i^t)|$$

with $C_s$: the current context, $C_t$: the context to compare, $p_i^s$ and $p_i^t$ perceptions of these contexts and $w_i$ the weight of those perceptions for this context. $w_i * |(p_i^s - p_i^t)|$ is bounded between 0 and 1/n, n is the number of perceptions. If a perception is not present a penalty equals to the maximum difference is imposed, its value is 1/n. This strategy is called a *pessimistic* measure in [15].

## 4 Contextual graphs

The main problem with the CxBR is that there is no way to express the temporal succession of actions for representing a complete strategy. To provide this feature we define Dynamic Contextual Graphs, based on Brézillon's contextual graphs [14].

Each Dynamic Contextual Graph (noted DCxG hereafter) may represent a different strategy or a different way to reach a goal. During the simulation, the succession of player's actions may correspond to a strategy. Our first goal is to determine which strategy is currently used by the player. Next, we will try to guide the learner in his actions, using the information contained in the contexts.

### 4.1 Dynamic Contextual Graphs

A DCxG is composed of an input context, an output context and an acyclic directed graph containing several nodes linked together by oriented arcs. To give a better view of DCxG, an example is described in figure 2. In this example, the goal of the strategy is to eliminate an opponent. We can see that two ways are proposed : the first one is to get help from a teammate and the second one is to dribble the opponent alone.

The input context ($Cin$ in figure 2) represents the context necessary to apply the strategy. The output context ($Cout$ in figure 2) is the context resulting from applying the strategy. An exception context may also be added to a DCxG to represent when a strategy should be aborted but this context is not used in this example.

In the acyclic graph representing the strategy, a node can be a contextual node, an action node, a cyclic action node or a sub-graph. A contextual node instantiates a context and is represented by a circle ($Cin$ to $C4$ in figure 2). Contextual nodes contain information about the necessary context for executing actions. Unlike in Brézillon's contextual graphs, instantiations of a contextual element are not associated with the arcs leaving the contextual node but are instantiated by the node itself.

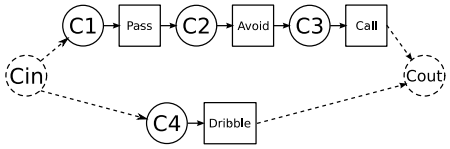


**Figure 2. The pass and go strategy represented by a DCxG**

Like in contextual graphs, an action is an executable method, meaning that an action can be done in a finite time. Action nodes may implement the actions available in the environment and are represented by boxes ($Pass$, $Avoid$, $Call$ and $Dribble$ in figure 2). The combination of instantiated contexts before an action defines the context needed to execute the action in the strategy : for example to Pass we need $\{Cin, C1\}$ and to Avoid only $\{C2\}$.

To keep the acyclic aspect of contextual graphs and to comply with dynamical environment, we defined a new kind of action : the cyclic action. This particular node represents an action repeated several times, the loop being

ended when the stop context associated to the action is closely similar to the personal context. In figure 2 for example, the action Avoid (the opponent) could have been noted : Run while opponent close.

The last type of node that could have been encountered is sub-graphs. Using sub-graphs in a DCxG allows the abstraction of actions. In figure 2 for example, the action Pass (to a teammate) could have been replaced by a sub-graph representing the succession of actions : Turn to be face to the teammate, Kick the ball.

### 4.2 Determining current strategy

DCxG will permit an expert to model several strategies. Using those graphs, we want to find the strategy matching the actions done by the learner. Our pointer system provide a way of comparing actions done by the learner to all the known strategies.

**Algorithm 1:** Following learner's actions

```
switch pointed node do
    case (Sub-)Graph beginning
        graphStack.push();
        New pointers on first nodes;
    end
    case (Sub-)Graph end
        graphStack.pop();
        Continue after sub-graph call or exit();
    end
    case Context
        contextStack.push();
        New pointers on next nodes;
    end
    case Action
        Wait action start;
        if action started == pointed action then
            Wait for action end;
            contextStack.clear();
            New pointers on next nodes;
        else
            Destroy pointer;
        end
    end
    case Calling a sub-graph
        Point on the sub-graph beginning;
    end
end
```

The pointers follow each possible path in each strategy. When the path does not correspond anymore to the actions done, the pointer destroys itself. Remaining pointers mark all the possible strategies.

Now, we are going to have a look at the algorithm describing the way a pointer follows a path, and how we can use the contexts' information to assess which strategy is best matching. The exploration of a graph by a pointer is described in algorithm 1. This algorithm is executed after each move of a pointer.

Each pointer has a context stack, containing the contexts necessary for the next action, and a graph stack storing the hierarchy of graphs the pointer entered.

Pointers move from node to node, following the oriented arcs. They duplicate themselves when several arcs leave a node, for the pointers to parse all the paths.

On each contextual node encountered, the context instantiated is pushed in a context stack associated to the pointer, and the pointer continues its exploration.The context stack is cleared after each action. When an action node is encountered, the pointer stops and waits for an action to be done by the learner. If the action done matches the action pointed, the pointer will continue to explore its path, else the pointer will destroy itself. This mechanism will eliminate all pointers that do not match the strategy used by the learner.

The positions of the remaining pointers in the graphs allow us to know the possible strategies that the learner is applying. Note that there may be several possibilities.

We use the context stack in order to assess the best strategy matching with the learner actions. Computing the similarity (cf. section 3.2) between the context stack and the personal context of the learner will give us a probability to associate to the strategy.

This algorithm can also provide a prediction of the learner's next action. When the pointers are stopped on action nodes, and before the learner does his action, we can use the probability computed from the context stack to determine which action should be done next.

## 5 Contextual help

Thanks to contextual graphs, we can set up two different types of help that we have already detailed in [16]. The first type allows a strong guidance because it comes before the learner acts. The second one is more weaker and allows to highlight relevant information to which learner did not pay

attention. The last one intervenes during action.

The first typeof help is done before the learner acts in the simulation. This help is based on an action prediction mechanism which is possible thanks to the contextual graph following algorithm. This algorithm allows the agent to predict the strategy and to know what should be the next step of this strategy. Thanks to this, it is possible to represent, in the virtual environment, the action that should be done. We have already defined some graphical primitives that are associated to the different actions. The example of the next part illustrates the setting up of the graphical primitive of the action *pass* in our environment.

The second type of help is based on contextual graphs, more precisely on contextual nodes. We are focusing on elements that learner should have not taken into account. In order to determine these elements we use our similarity measurement to find the difference between the current context and the expected one. The absence or the change of a perception allows the agent to know what elements have not been taken into account. It is possible to highlight them in the simulation. As for actions, a graphical primitive can be associated to a perception thanks to the *pedagogical_feature* that have been defined in every type of perception.

The example of the next section shows the setting up of the assistance that is done before action, but we could have illustrated the help during action on this example and notice that the learner did not see its partner. The result should have been the enlarging of its partner, which is the pedagogical primitive of the perception *presence*.

## 6 An example of assistance

In this example, we are focusing on the help done before action. We are considering that the player *A1* is a learner and he can see two partners but only one of them is unmarked. The agent *A1* point of view on this situation is shown on figure 3, players *A2* and *A3* are its partners whereas *B1* is its opponent.

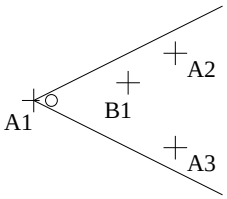


**Figure 3. Situation representation**

Figure 4 shows a little part of the dynamic contextual graphs library of this agent. One can see that the remaining graph is the one introduced in section 4 represented by figure 2 and called *pass and go*. We did not developed the entry context of each graph of the library, but as we have already said it is with this context that agent decides to keep a graph or not. We are focusing, in this example, on the fork that one can see on figure 4.

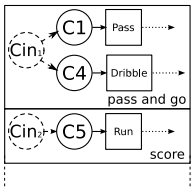


**Figure 4. A part of DCxG library**

The action that should be executed to follow the strategy strongly depends on the current context. The agent will compare current context with $C4$ and $C1$ to determine the most similar one. Thanks to the following graph algorithm introduced in section 4, it can determine what should be the next action and it can highlight it. In this strategy, the difference between the two contexts is the presence of an unmarked partner in $C1$. In the case of our example, a partner is present (*A3*) and unmarked, so the most similar context is $C1$. The agent will highlight the action located just after the contextual node $C1$, here it is a *pass*.

Figure 5 shows this help, one more time, context information is used to set the parameters of the graphical primitive associated to this action, here it is a curved and oriented arrow. In this case, it consists in orienting arrow towards the player *A3*.

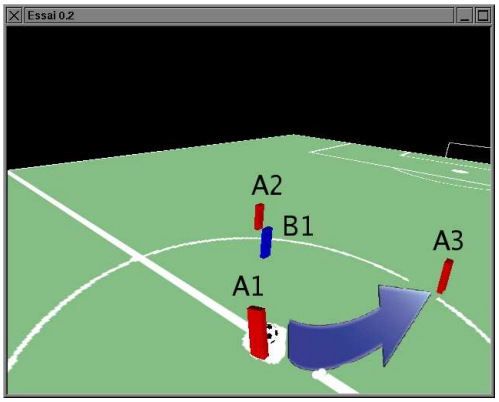


**Figure 5. Contextual help**

## 7 Conclusion

The aim of this article is to show that the context is an information medium. We have illustrated the different ways we use context in virtual environment for training. On one side, it allows to set up a decision-making engine for autonomous agent in an environment, context then offers a filter for agent perceptions. On the other side, it allows to share information between a learner and pedagogical agents of the simulation.

This last point implies the possibility of reusing information given by an expert for certain situations because we can not describe a dynamic situation as a static scenario, but only as strategies being the parts of the final scenario.

The example highlights the use of contextual graphs for setting up an assistance. This type of help is context dependent. Using context allows to make relevant elements for decision-making explicit. We are currently working with psychologists to build a case base and to prepare a protocol to validate the explanations generation.

## References


[1] J. Piaget and B. Inhelder. *The Psychology of the Child*. NY: Basic Books, 1969.

[2] J. Rickel and W. Lewis Johnson. Steve (video session): a pedagogical agent for virtual reality. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 332–333, New York, NY, USA, 1998. ACM Press.

[3] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier. Multiagents systems for virtual environment for training. application to fire-fighting. *Special issue "Advanced Technology for Learning" of International Journal of Computers and Applications (IJCA)*, 1(1):25–34, juin 2004.

[4] D. M. Povovici, J. P. Gerval, P. Chevaillier, J. Tisseau, J. Serbanati, and P. Guegen. Educative distributed virtual environments for children. *Journal of Distance Eduction Technologies*, 2(4):18–40, 2004.

[5] L. Rognin, P. Salembier, and M. Zouinar. Cooperation, interactions and socio-technical reliability: the case of air-traffic control. comparing french and irish settings. In *Proceedings of the European Conference on Cognitive Ergonomics (ECCE9)*, 1998.

[6] J-M Cellier and J-M Hoc. La gestion d'environnements dynamiques. *Psychologie Francaise*, 46:103–106, 2001.

[7] M.T. Argilaga and G.K. Jonsson. Detection of real-time patterns in sports: Interactions in soccer. *International Journal of Computer Science in Sport*, Volume 2/Edition 2(1):118–121, 2003.

[8] D. D. Woods. Coping with complexity: the psychology of human behaviour in complex systems. *Tasks, errors, and mental models*, pages 128–148, 1988.

[9] A. J. Gonzalez and R. Ahlers. Context-based representation of intelligent behavior in training simulations. *Trans. Soc. Comput. Simul. Int.*, 15(4):153–166, 1998.

[10] P. Brézillon and J. Pomerol. Contextual knowledge sharing and cooperation in intelligent assistant systems, 1999.

[11] J-Ch. Pomerol and P. Brézillon. About some relationships between knowledge and context. In Varol Akman, Paolo Bouquet, Richmond Thomason, and Roger A. Young, editors, *Modeling and Using Context: Third International and Interdisciplinary Conference, Context 2001*, pages 461–464, Berlin, 2001. Springer-Verlag.

[12] R. Bénard, C. Bossard, and P. De Loor. Context's modelling for participative simulation. In *19th International Florida Artificial Intelligence Research Society Conference (FLAIRS-06)*. AAAI, 2006.

[13] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.

[14] P. Brézillon. Context dynamic and explanation in contextual graphs. In Patrick Blackburn, Chiara Ghidini, Roy M. Turner, and Fausto Giunchiglia, editors, *Modeling and Using Context: Fourth International and Interdisciplinary Conference, Context 2003*, pages 94–106, Berlin, 2003. Springer-Verlag.

[15] S. Bogaerts and D. B. Leake. Facilitating cbr for incompletely-described cases: Distance metrics for partial problem descriptions. In *ECCBR*, pages 62–76, 2004.

[16] R.Bénard, P. De Loor, and J. Tisseau. Understanding dynamic situations through context explanation. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies, ICALT*, 2006.